WHAT MAKES CROSS-VARIABLE MIXING EFFECTIVE? VARIABLE-SPECIFIC VALUE INTERFACES FOR MULTIVARIATE TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent multivariate time-series forecasting methods increasingly exploit cross-variable dependencies through mixing mechanisms that determine which variables communicate and how strongly. However, variables often differ in periodicity, phase, lag, and predictive role. Selecting a useful source therefore does not determine how its information should be expressed for different targets, yet existing mixers typically aggregate values in a shared latent space. We revisit cross-variable aggregation through variable-specific value interfaces. Our FRAMEMIX operator equips variables with learnable orthogonal value interfaces for expressing outgoing information and interpreting aggregated information. Composing the source and receiver interfaces yields a distinct value transformation for every ordered variable pair without directly learning pairwise operators. We incorporate FrameMix into FRAMEFORMER, a patch-based architecture that mixes variables at each patch position before modeling temporal dependencies within each variable. Experiments show that variable-specific interfaces improve multiple routing mechanisms and remain effective with both learned and fixed frames, indicating that their gains cannot be explained by additional trainable capacity alone. More expressive pair-specific alternatives require substantially more parameters but provide no reliable accuracy gains. Frameformer achieves the lowest error on six of eight benchmarks at horizon 96 and is most effective on strongly coupled multivariate data, while results at horizons 192 to 720 on six benchmarks are mixed.

1 INTRODUCTION

Multivariate time-series (MVTS) forecasting increasingly relies on cross-variable mechanisms—including variate attention, learned graphs, and explicit cross-dimension stages—to exchange information across variables (Liu et al., 2024; Wu et al., 2020; Cao et al., 2020; Zhang & Yan, 2023). Yet the benefit of such mixing remains inconsistent. Channel-independent models often match or outperform architectures designed for cross-variable interaction (Zeng et al., 2023; Nie et al., 2023), suggesting that explicitly modeling inter-variable dependencies is not always beneficial. However, this pattern is not universal. In many forecasting tasks with strongly coupled variables, removing cross-variable communication substantially degrades performance. These seemingly conflicting findings suggest that the central question is not simply whether cross-variable mixing helps, but which component of the mixing mechanism carries its predictive benefit when mixing is useful.

Most existing methods answer this question through what we call *routing*: the scalar cross-variable weights, such as attention scores or graph edges, that determine which variables communicate and how strongly each source contributes. Attention weights, graph edges, and related routing mechanisms select the information received by each variable, while the communicated values are typically processed in a representation space shared across all variables. This implicitly assumes that, once a useful source has been selected, its representation can be delivered to different receivers in essentially the same form.

However, variables in multivariate time series often differ in physical meaning, periodicity, phase, scale, lag, and predictive role. As a result, information from the same source variable can play differ-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

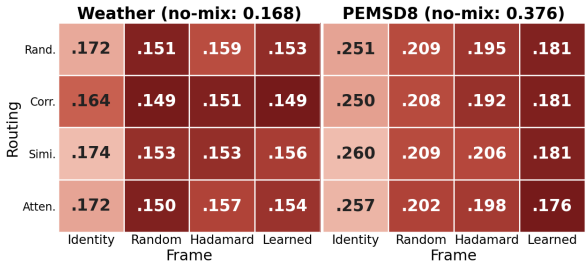


Figure 1: Variable-specific frames consistently improve forecasting performance across diverse routing mechanisms (lower/darker is better). Replacing shared-space mixing (Identity) with any non-identity frame reduces error for nearly every routing rule on both datasets. Random denotes frozen, randomly initialized Householder frames with no trainable frame parameters. Values in parentheses indicate the channel-independent (no cross-variable mixing) baseline.

ent roles for different receivers. For example, a traffic sensor may provide an early warning for one road segment, reflect congestion at another with a delay, and be largely irrelevant to a third. Consequently, the same source information may need to be expressed differently for different receivers. This holds even after all variables are embedded into a common latent space: a shared embedding makes values comparable, but it applies the same map to every variable and cannot undo receiver-dependent differences such as a relative lag, which for a periodic component is a rotation of its phase coordinates (Appendix H.1). Routing determines *who communicates* and *with what strength*, but it does not determine *how communicated values should be expressed by the source and interpreted by the receiver*. We call this second design axis a *variable-specific value interface*.

Motivated by this perspective, we introduce FRAMEMIX, a cross-variable mixing operator that realizes variable-specific value interfaces through learnable orthogonal frames. Each variable owns a frame that expresses its outgoing values and interprets incoming aggregated information: intuitively, each source translates its message into a common exchange format before mixing, and each receiver translates the mixture back into its own format. Composing the source and receiver frames produces an effective transformation for every ordered source–receiver pair, enabling pair-specific value processing without requiring an independent transformation for every variable pair while leaving the routing mechanism unchanged. Figure 1 illustrates the central observation motivating our approach: replacing conventional shared-space mixing with variable-specific frames consistently improves forecasting performance across diverse routing mechanisms.

To isolate the role of value interfaces, we compare matched models that share the backbone and routing and differ only in their frames; identity frames exactly recover shared-space mixing. Non-identity frames consistently improve accuracy across routing rules, including where shared-space mixing gives little or no benefit, and fixed random Householder and Hadamard frames already achieve much of the gain, whereas far more expressive pair-specific parameterizations add little. Permuting routing assignments during training costs little, whereas breaking the correspondence between variables and frames removes most of the frame gain (Appendices B.2 and C). Under a common training protocol, Frameformer achieves the lowest mean error on six of eight benchmarks with far fewer parameters than the heaviest cross-variable baselines, Crossformer and TimeXer. Our contributions are: (i) we separate cross-variable aggregation into routing and variable-specific value interfaces and show that the latter gives gains beyond routing alone; (ii) we introduce FrameMix, which realizes these interfaces with orthogonal frames and pair-specific value processing without pair-specific parameters, within the patch-based FRAMEFORMER; (iii) through controlled analyses of frame presence, construction, trainability, ownership, placement, and parameterization, we show that the gains come from persistent variable-specific interfaces rather than added capacity.

2 RELATED WORK

Multivariate time-series forecasting. After early Transformer forecasters focused on the temporal axis (Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022; Wu et al., 2023), recent methods differ mainly in how they treat the variable dimension. Channel-independent methods such as DLin-

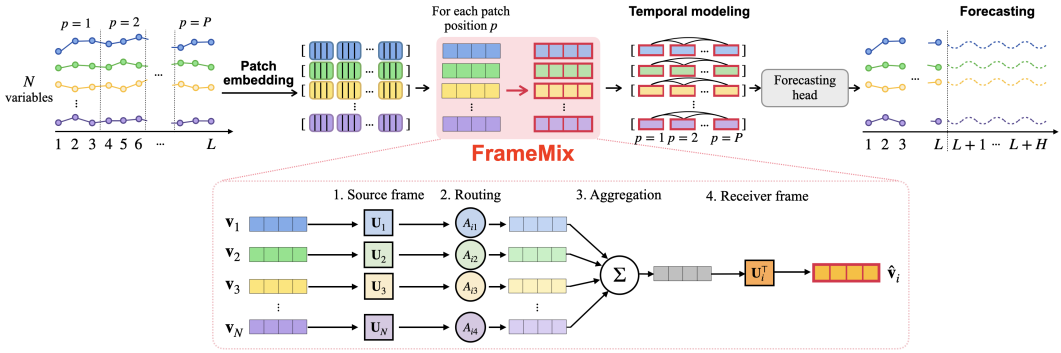


Figure 2: An overview of the Frameformer.

ear (Zeng et al., 2023), PatchTST (Nie et al., 2023), and FITS (Xu et al., 2024) process each variable separately, trading capacity for robustness (Han et al., 2024b). Cross-variable methods instead exchange information through attention, learned graphs, or mixing stages. iTransformer (Liu et al., 2024) applies attention across variable tokens, Crossformer (Zhang & Yan, 2023) introduces a dedicated cross-dimension stage, TSMixer (Chen et al., 2023) and TimeMixer (Wang et al., 2024a) mix across channels with MLPs, SOFTS (Han et al., 2024a) fuses all variables into a global core representation and redistributes it, Timer-XL (Liu et al., 2025) attends jointly over variables and time, CCM (Chen et al., 2024) shares parameters within channel clusters, and graph-based forecasters (Wu et al., 2019; 2020; Cao et al., 2020; Yi et al., 2023) learn inter-variable adjacencies for message passing. Recent work routes channel–patch dependencies through frequency experts (xCPD; Li et al., 2026), models fine-grained interactions with channel-specific kernels (FACT; Wang et al.), makes interaction permutation-invariant for unseen channels (CPiRi; Xu et al., 2026), or handles asynchrony and missingness (Jang et al., 2026). Many of these methods share value processing across variables, FACT being an exception; FrameMix instead isolates one component, adding only persistent per-variable interfaces on the value path of an otherwise unchanged host, so that it can be tested in controlled interventions.

Among cross-variable methods, interaction has been formulated primarily through routing: determining which variables should communicate and how strongly. Accordingly, prior comparisons have largely focused on whether cross-variable mixing is beneficial or which routing mechanism performs best. We instead decompose cross-variable aggregation into two components: routing, which selects and combines sources, and variable-specific value interfaces, which determine how information is expressed by each source and interpreted by each receiver. Several recent methods implicitly modify how communicated values are processed, suggesting that this second component may influence forecasting performance. CARD (Zhou et al., 2024) introduces channel-aligned attention, LIFT (Zhao & Shen, 2024) explicitly adjusts lead–lag relationships, and CI/mixing hybrids such as CSformer (Wang et al., 2025) add lightweight cross-variable paths to channel-independent backbones. Because these methods alter multiple aspects of the forecasting architecture together, however, their predictive gains do not isolate routing, additional capacity, variable-specific processing, or persistent variable–parameter correspondence. Our matched interventions isolate these factors through controlled comparisons, allowing the contribution of variable-specific value interfaces to be isolated from routing and model capacity.

Sheaf-based transport. Sheaf neural networks associate local representation spaces with graph entities and use linear maps to transport information between them. Sheaf attention (Barbero et al., 2022b) and connection-Laplacian methods (Barbero et al., 2022a) incorporate such transport into graph propagation, while Bundle Neural Networks (Bamberger et al., 2025) assign orthogonal maps to nodes and use the resulting flat-bundle structure for graph diffusion. PolyNSD (Borgi et al., 2025) uses polynomial spectral filters over the sheaf Laplacian to obtain multi-hop propagation even with diagonal restriction maps. Concurrent work on temporal link prediction (Asif et al., 2026) uses event-updated Householder frames to transport node memories between local coordinate systems. Householder products are a standard orthogonal parameterization in recurrent networks (Mhammedi

et al., 2017), alongside Cayley and exponential maps (Helfrich et al., 2018; Lezcano-Casado & Martınez-Rubio, 2019).

The orthogonal node-frame construction itself is therefore not new; our contribution is to place it on the value path of multivariate cross-variable mixing, separate it from routing, and dissect what carries its benefit.

3 Our Approach

We present *Frameformer*, a patch-based forecaster that improves cross-variable aggregation through variable-specific value interfaces (Figure 2). Each variable is embedded as a sequence of patch tokens; at each patch position, variables exchange information through FrameMix, after which a channel-independent temporal Transformer and a forecasting head produce the forecast.

Problem setup. Let $\mathbf{X} \in \mathbb{R}^{L\times N}$ denote a multivariate time series containing N variables observed over the past L time steps. The forecasting task is to predict the next H time steps, $\mathbf{Y} \in \mathbb{R}^{H\times N}$.

A patch-based forecaster partitions the history of each variable into P temporal patches and embeds them into latent tokens. Let $\mathbf{v}_i^{(p)} \in \mathbb{R}^d$ denote the value representation of variable i at patch position p. At each patch position, a cross-variable mixer produces an updated representation $\widehat{\mathbf{v}}_i^{(p)}$ by aggregating information from the other variables. Temporal modeling is then performed across patch positions within each variable before the forecasting head generates the final predictions.

Our focus is the cross-variable aggregation operator. We modify how value representations are exchanged across variables while leaving the temporal modeling and forecasting unchanged.

Routing and value interfaces. In multivariate time series, variables exhibit heterogeneity: they differ in periodicity, phase, and lag structure. These differences may require their latent representations to be expressed differently when information is exchanged across variables, whereas conventional cross-variable aggregation typically combines them in a shared representation space. Scalar routing weights can control the magnitude of each source contribution but cannot change how that contribution is represented. We accordingly write cross-variable aggregation as

$$\widehat{\mathbf{v}}_i = \mathbf{T}_i \sum_{j=1}^{N} A_{ij}\mathbf{S}_j\mathbf{v}_j, \tag{1}$$

where A_{ij} is the scalar routing weight from source variable j to receiver variable i, $\mathbf{S}_j \in \mathbb{R}^{d\times d}$ is a source-specific value map, and $\mathbf{T}_i \in \mathbb{R}^{d\times d}$ is a receiver-specific value map.

We use *routing* to refer broadly to the mechanism that determines the weights A_{ij}, including fixed rules, similarity or correlation scores, static learned mixing matrices, and learned attention rules. Routing therefore specifies which sources contribute and how strongly. In contrast, the value maps specify the coordinate convention in which a source expresses its information and the form in which a receiver interprets the resulting mixture.

Existing cross-variable mixers differ mainly in how the pairwise weights are obtained: learned attention over variate tokens, a static learned channel-mixing matrix, or a correlation-based graph. In many commonly used mixers, however, values are aggregated through a shared value representation between variables and do not contribute pair specificity, which corresponds to the setting $\mathbf{S}_j = \mathbf{T}_i = \mathbf{I}$ and yields

$$\widehat{\mathbf{v}}_i = \sum_{j=1}^{N} A_{ij}\mathbf{v}_j. \tag{2}$$

In this case, ordered-pair specificity enters only through the scalar weight A_{ij}, which applies the same rescaling to every coordinate and cannot provide receiver-specific coordinate-wise processing.

By introducing source and receiver interfaces, Eq. (1) adds a second form of pair specificity: the composed map $\mathbf{T}_i\mathbf{S}_j$ determines how source j's contribution is re-expressed for receiver i. Because this pairwise map is composed from one interface per variable, it enables source–receiver-specific

processing without learning an independent transformation for every ordered pair. Appendix H.1 characterizes when this matters: shared-space mixing suffices only when the ideal transfer is common to all pairs, whereas node-factorized orthogonal interfaces are exact when variables differ by relative rotations such as phase lags; on synthetic data with known lags, learned frames reduce error by 18% only when the lags differ (Appendix H.8). We next instantiate these interfaces using variable-specific orthogonal frames.

The FrameMix operator. We propose FrameMix, which realizes variable-specific value interfaces by assigning each variable a learnable orthogonal frame. Specifically, Eq. (1) is instantiated by setting $\mathbf{S}_j = \mathbf{U}_j$ and $\mathbf{T}_i = \mathbf{U}_i^\top$, yielding

$$\begin{aligned}\widehat{\mathbf{v}}_i &:= \mathrm{FrameMix}_i\left(\{\mathbf{v}_j\}_{j=1}^N; \mathbf{A}, \mathcal{U}\right) \\ &= \mathbf{U}_i^\top \sum_{j=1}^N A_{ij}\, \mathbf{U}_j\, \mathbf{v}_j = \sum_{j=1}^N A_{ij} \underbrace{\left(\mathbf{U}_i^\top \mathbf{U}_j\right)}_{\mathbf{R}_{ij}} \mathbf{v}_j.\end{aligned} \tag{3}$$

where $\mathcal{U} = \{\mathbf{U}_i\}_{i=1}^N$ and $\mathbf{R}_{ij} = \mathbf{U}_i^\top \mathbf{U}_j$ is the effective value transformation associated with the ordered source–receiver pair (i, j).

The routing weight A_{ij} determines how strongly source variable j contributes, whereas $\mathbf{R}_{ij}$ determines how that contribution is represented for receiver variable i. Since each pairwise transformation is constructed from the corresponding source and receiver frames, FrameMix realizes an effective transformation for every ordered source–receiver pair while learning only one frame per variable. These transformations are coupled rather than independent: $\mathbf{R}_{ii} = \mathbf{I}$, $\mathbf{R}_{ji} = \mathbf{R}_{ij}^\top$, and $\mathbf{R}_{ik}\mathbf{R}_{kj} = \mathbf{R}_{ij}$. In Frameformer the frames are additionally modulated across patch positions (Eq. (6)). FrameMix modifies only the value path and leaves the routing weights unchanged.

Each frame is parameterized as a product of k Householder reflections,

$$\mathbf{U}_i = \prod_{\ell=1}^k \mathcal{H}(\mathbf{h}_{i,\ell}), \qquad \mathcal{H}(\mathbf{h}) = \mathbf{I} - \frac{2\mathbf{h}\mathbf{h}^\top}{\mathbf{h}^\top \mathbf{h}}. \tag{4}$$

Here, $\mathbf{h}_{i,\ell} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ is the learnable vector parameterizing the ℓ-th Householder reflection for variable i. This parameterization guarantees orthogonality, $\mathbf{U}_i^\top \mathbf{U}_i = \mathbf{I}$, while requiring $O(kd)$ parameters per variable; the vectors are initialized i.i.d. Gaussian, so each frame starts as a random product of k reflections.

Frameformer architecture. Our Frameformer places FrameMix between patch embedding and temporal modeling. Given a multivariate time series, each variable is first converted into a sequence of patch tokens. FrameMix then performs cross-variable aggregation independently at each patch position. The resulting tokens are processed across patch positions within each variable, and a forecasting head maps the temporal representations to the prediction horizon.

Patch embedding. For each variable i, the input history $\mathbf{x}_i \in \mathbb{R}^L$ is divided into P non-overlapping patches of length s, where $P = L/s$. A shared linear projection maps the p-th patch to a d-dimensional token:

$$\mathbf{v}_i^{(p)} = \mathbf{W}_{\mathrm{patch}} \mathbf{x}_i^{(p)} + \mathbf{e}^{(p)}, \tag{5}$$

where $\mathbf{x}_i^{(p)} \in \mathbb{R}^s$ is the p-th patch of variable i and $\mathbf{e}^{(p)}$ is its positional embedding.

Cross-variable FrameMix. At each patch position p, FrameMix is applied across the N variable tokens:

$$\widehat{\mathbf{v}}_i^{(p)} = \mathrm{FrameMix}_i\left(\{\mathbf{v}_j^{(p)}\}_{j=1}^N; \mathbf{A}^{(p)}, \mathcal{U}^{(p)}\right), \tag{6}$$

where $\mathbf{A}^{(p)} = [A_{ij}^{(p)}]_{i,j=1}^N$ is the routing-weight matrix and $\mathcal{U}^{(p)} = \{\mathbf{U}_i^{(p)}\}_{i=1}^N$ is the set of variable-specific frames. In Frameformer, $A_{ij}^{(p)}$ is a parameter-free scaled dot-product softmax over the layer-normalized tokens at patch p. Frames are not independent per patch: they share one set of vectors per variable with a smooth gated modulation over patch positions, giving $2kd + 1$ frame parameters per variable regardless of P (Appendix H.3).

	Identity	Random	Hadamard	FrameMix
Weather	.172	**.150**	.157	.154
Solar	.192	.205	.179	**.179**
Electricity	.153	.144	.144	**.140**
PEMSD8	.257	.202	.198	**.176**
PEMSD4	.212	.152	.144	**.111**

Table 1: Test MSE of Frameformer and its variants using different frames. Random: frozen random Householder frames.

	Identity	Affine	FrameMix
Weather	.172	.157	**.154**
Solar	.192	.187	**.179**
Electricity	.153	**.139**	.140
PEMSD8	.257	.181	**.176**
PEMSD4	.212	.120	**.111**

Table 2: Test MSE of Frameformer (orthogonal FrameMix) and its non-orthogonal affine-interface variant.

Temporal modeling. For each variable i, the FrameMix outputs across patch positions are stacked into $\widehat{\mathbf{V}}_i = [\widehat{\mathbf{v}}_i^{(1)}, \ldots, \widehat{\mathbf{v}}_i^{(P)}]^\top$. A channel-independent temporal Transformer (Vaswani et al., 2017) processes this sequence across patches, producing $\mathbf{Z}_i$.

Forecasting head. A shared forecasting head maps $\mathbf{Z}_i$ to $\widehat{\mathbf{y}}_i \in \mathbb{R}^H$, and the variable-wise predictions are collected as $\widehat{\mathbf{Y}} = [\widehat{\mathbf{y}}_1, \ldots, \widehat{\mathbf{y}}_N] \in \mathbb{R}^{H \times N}$.

Training objective. Frameformer is trained end-to-end with the mean squared error:

$$\mathcal{L} = \frac{1}{HN} \sum_{t=1}^{H} \sum_{i=1}^{N} \left(Y_{t,i} - \widehat{Y}_{t,i} \right)^2 . \tag{7}$$

4 Analyzing Cross-Variable Value Interfaces

Section 3 separates cross-variable aggregation into routing and value interfaces. In this section, we first measure the stagewise contribution of each component, then test whether the gains from frames persist across different routing and frame choices, and finally examine whether orthogonality itself is necessary for these gains.

Routing and frames make stagewise contributions. We evaluate a nested within-host ablation with three cumulative conditions. (i) *Backbone* disables cross-variable aggregation ($\mathbf{S}_j = \mathbf{T}_i = \mathbf{I}$, $A_{ij} = \delta_{ij}$). (ii) *Backbone + routing* adds attention-based cross-variable aggregation while retaining the shared value space ($\mathbf{S}_j = \mathbf{T}_i = \mathbf{I}$). (iii) *Backbone + routing + frame* further adds variable-specific frames to the same routed mixer.

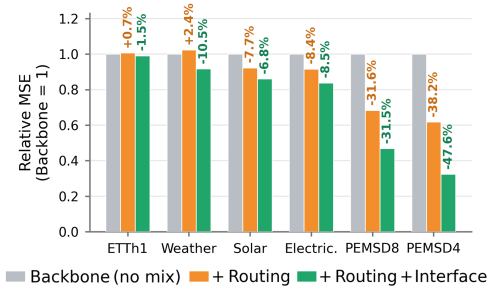


Figure 3: Stagewise gains from routing and interfaces (MSE / no-mixing backbone; lower is better).

In Figure 3, routing reduces MSE by 7.7%–8.4% on Solar and Electricity and frames add a further 6.8%–8.5%; on PEMSD8 and PEMSD4 routing gives 31.6% and 38.2% and frames a further 31.5% and 47.6%. On Weather, routing alone slightly increases error, yet adding frames makes the routed model better than the backbone; ETTh1, a negative control, changes little. Frames thus neither replace nor duplicate routing.

Frame gains persist across routing and frame choices. The stagewise comparison above uses a dot-product attention router and learned frames. We next test whether the frame gain depends on either of these two choices.

Figure 1 compares shared-space aggregation with fixed random Householder, fixed Hadamard, and learned frames under four routing rules: a fixed random stochastic matrix, row-normalized absolute correlation of the training series, a cosine-similarity softmax, and dot-product attention

Model	ETTh1	ETTh2	Exch.	Weather	Solar	PEMSD8	PEMSD4	Electric.
DLinear	.463±.003	.244±.004	**.106±.003**	.198±.002	.283±.001	.652±.008	.425±.001	.196±.000
PatchTST	.467±.010	.250±.004	.115±.006	.176±.003	.218±.017	.435±.003	.507±.015	.170±.003
iTransformer	.483±.016	.248±.001	.114±.001	.176±.002	.206±.002	.376±.008	.322±.004	.154±.000
TimeXer	.465±.006	.240±.002	.113±.002	.164±.002	.199±.008	.269±.018	.158±.001	.142±.002
Crossformer	.467±.003	.473±.081	1.075±.166	.165±.002	.194±.009	.224±.005	.112±.003	.158±.009
CARD	.496±.017	.276±.010	.155±.015	.163±.003	.208±.005	.265±.031	.182±.012	.146±.001
CSformer	.470±.007	**.234±.003**	.110±.004	.167±.005	.203±.007	.358±.003	.285±.028	.148±.003
Frameformer	**.455±.004**	.235±.002	.115±.007	**.154±.002**	**.179±.005**	**.176±.002**	**.111±.001**	**.140±.003**

Table 3: Test MSE on eight multivariate forecasting benchmarks at $h = 96$ (3-seed mean $\pm$ standard deviation). Bold marks the lowest mean in each column.

(Appendix H.7). Across the routing rules considered, every non-identity frame improves over its matched shared-space counterpart. On Weather, frames make mostly ineffective routers beneficial. On PEMSD8, they add further gains to routers that already help.

Table 1 extends the construction comparison across additional datasets. Hadamard improves over Identity on every displayed dataset, while the random frame does so on all but Solar. Learned frames provide their clearest additional gains on PEMSD8 and PEMSD4. Thus, the frame benefit is not tied to one learned router or to additional trainable frame parameters. Fixed frames already give each variable a persistent, distinct convention to which the shared downstream modules co-adapt (Appendix D); learning the frames adds the largest further gains on the strongly coupled PEMS datasets. Ownership matters more than routing: reassigning frames across variables at test time raises PEMSD4 error from .111 to .650 (Appendix C.1), whereas training with shuffled routing weights stays within 2% of normal accuracy on all six datasets (Appendix B.2).

Are orthogonal frames necessary? To test whether the gain requires orthogonality, we replace the orthogonal maps with per-variable affine functions, $\mathbf{S}_j(\mathbf{v}) = \mathbf{s}_j^{\mathrm{src}} \odot \mathbf{v} + \mathbf{b}_j^{\mathrm{src}}$ and $\mathbf{T}_i(\mathbf{z}) = \mathbf{s}_i^{\mathrm{tgt}} \odot \mathbf{z} + \mathbf{b}_i^{\mathrm{tgt}}$, while preserving the same routing weights and value-path placement.

As shown in Table 2, the affine variant recovers 91%–94% of the Identity-to-FrameMix improvement on the PEMS datasets and 83% on Weather. It slightly outperforms orthogonal FrameMix on Electricity, whereas on Solar it recovers only 38% of the gain. These results indicate that variable-specific interfaces account for much of the improvement in several datasets, while orthogonal structure provides additional benefits in selected regimes. FrameMix remains a useful instantiation: it preserves value norms, ties both maps through one frame per variable, and is best among these variants on four of the five datasets.

5 Experiments

Setup. We evaluate Frameformer on eight benchmarks (ETTh1, ETTh2, Exchange, Weather, Solar, PEMSD8, PEMSD4, and Electricity) at horizon $h = 96$, and six of them at $h \in \{192, 336, 720\}$, and compare it with DLinear (Zeng et al., 2023), PatchTST (Nie et al., 2023), iTransformer (Liu et al., 2024), TimeXer (Wang et al., 2024b), Crossformer (Zhang & Yan, 2023), CARD (Zhou et al., 2024), and CSformer (Wang et al., 2025). All methods share the same data loaders, splits, and training-set normalization and are trained with Adam (learning rate 10^{-3}, MSE loss, at most 15 epochs with early stopping). Baselines follow their papers (CSformer, lacking public code, is reproduced from its description); this is a common-protocol comparison, not per-model tuning. Frameformer uses $d_{\mathrm{model}} = 128$, four Householder reflections per variable, two Transformer blocks, and a patch size of 16 (details in Appendix A). We report the three-seed mean and standard deviation.

Main results. As shown in Table 3, Frameformer achieves the lowest mean MSE on six of the eight benchmarks and ranks second on ETTh2, trailing the best mean by only 0.001. It ranks first on both PEMSD8 and PEMSD4, which are also the datasets where the controlled interface comparisons show the largest improvements over identity-based aggregation. On PEMSD4, Frameformer reaches 0.111 MSE with approximately 0.79M parameters, compared with Crossformer's 0.112 using approximately 9.3M parameters, and one training step takes 37 ms versus 97–115 ms for Cross-

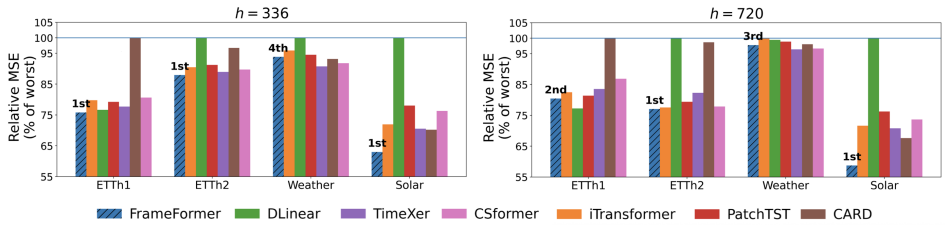


Figure 4: Forecasting performance at $h = 336$ and $h = 720$. MSE is normalized by the worst-performing method, so 100% denotes the worst result and lower is better.

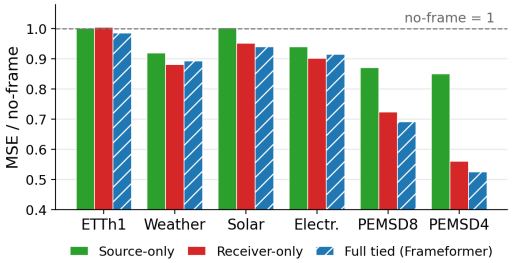


Figure 5: Value-path frame placement (test MSE relative to no-frame = 1; 3 seeds).

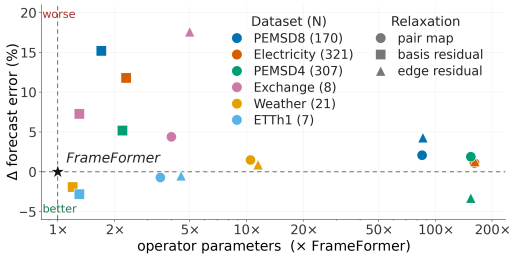


Figure 6: Accuracy–capacity trade-off of value-interface relaxations. Lower y is better. x shows interface parameters relative to Frameformer on a log scale.

former, CARD, and CSformer (Appendix H.4). Frameformer outperforms iTransformer, DLinear, and TimeXer on seven of the eight benchmarks, and outperforms PatchTST on seven with one tie.

Multi-horizon evaluation. Figure 4 covers $h \in \{336, 720\}$ on four datasets.

Frameformer ranks first on ETTh2 and Solar at both horizons, and first or second on ETTh1. Its advantage is especially pronounced on Solar, where it remains substantially below all competing methods at both horizons. Weather is more tightly clustered: Frameformer ranks fourth at $h = 336$ and third at $h = 720$, with only small absolute differences among the leading methods. Appendix F reports all baselines at $h \in \{192, 336, 720\}$ on six datasets; on PEMSD8 and PEMSD4 we report Frameformer's horizon curve only.

Operating regime. Consistent with the controlled analyses, Frameformer is strongest on the coupled sensor datasets, where routing helps and interfaces add substantial gains. Exchange provides the clearest boundary: DLinear obtains an MSE of 0.106, whereas Frameformer reaches 0.115. This result suggests that variable-specific interfaces are most beneficial when cross-variable mixing itself provides useful predictive information.

6 Additional Analysis

We now return to Eq. (1) to isolate source and receiver interfaces and to test whether independent pair-specific operators add useful capacity.

Where does the interface gain arise? We next ask which component drives the interface gain: source-specific expression, receiver-specific interpretation, or their combination. Starting from shared-space mixing (no-frame) $\widehat{\mathbf{v}}_i = \sum_j A_{ij}\mathbf{v}_j$, we compare a source-only interface $\widehat{\mathbf{v}}_i = \sum_j A_{ij}\mathbf{U}_j\mathbf{v}_j$, a receiver-only interface $\widehat{\mathbf{v}}_i = \mathbf{U}_i^\top \sum_j A_{ij}\mathbf{v}_j$, and the full tied interface Frameformer $\widehat{\mathbf{v}}_i = \mathbf{U}_i^\top \sum_j A_{ij}\mathbf{U}_j\mathbf{v}_j$. The encoder, router, temporal stage, decoder, data splits, and optimization are otherwise fixed.

Figure 5 shows that receiver-specific interpretation accounts for most of the interface gain. The receiver-only interface reduces MSE by 4.8% on Solar, 27.6% on PEMSD8, and 43.9% on PEMSD4, recovering most of the improvement obtained by the full tied interface. It also slightly outperforms the full composition on Weather and Electricity. The source-only interface is weaker than the receiver-only one on five of six datasets and gives no gain on Solar.

Source-specific expression nevertheless provides a complementary gain in strongly coupled regimes. The full tied interface improves over the receiver-only interface on Solar, PEMSD8, and PEMSD4, by approximately 4.1% on average across these datasets, and achieves the lowest MSE on four of the six datasets overall. Thus, receiver-specific interpretation is the primary contributor, while source-specific expression provides additional value when cross-variable coupling is strong. We attribute this to the shared patch embedding, which already aligns sources, and to the receiver map being the only variable-specific parameter before the shared temporal encoder and head, whereas a source-only map mixes contributions in inconsistent conventions (Appendix H.2).

Do variable-specific interfaces suffice for pairwise value transport? Frameformer constructs each ordered-pair value transformation as $\mathbf{R}_{ij} = \mathbf{U}_i^\top \mathbf{U}_j$, using one k-reflection frame per variable and $O(Nkd)$ interface parameters. We test whether this factorization sacrifices expressivity with three relaxations, holding the encoder, router, and decoder fixed: an independent dense *pair map* per ordered pair, a *basis residual* with pair-specific coefficients over a shared low-rank basis (full N), and an independent low-rank *edge residual* per pair. The two quadratic variants use $N = 25$ on PEMSD4, PEMSD8, and Electricity and full N elsewhere, each against the matched Frameformer.

As illustrated in Figure 6, none of the relaxations provides a consistent accuracy gain. At full N, the basis residual increases MSE by 5.2%–15.2% on PEMSD4, Exchange, Electricity, and PEMSD8, while improving Weather and ETTh1 by only 2%–3%. The more expensive pair-specific alternatives also mostly increase error. Their only notable gain is a 3.3% improvement from the edge residual on PEMSD4, measured at $N = 25$; at full N it would need $154\times$ the interface parameters. At full N, the relaxations would need up to $162\times$ the interface parameters (53M versus 329k on Electricity) without a reliable benefit. Thus, $\mathbf{U}_i^\top \mathbf{U}_j$ is not merely a compressed approximation to independent pairwise operators, but a useful inductive bias. Accuracy is flat for $k \in [2, 8]$ reflections, and learned frames partly align with series correlations (Appendices H.5, H.6).

Does the lag model predict when interfaces help? On synthetic data where twelve variables are lagged noisy copies of one driver (Appendix H.8), we compare synchronous copies, for which the ideal transfer is common to all pairs, with lags spread over 0–60 steps, for which it depends on the pair. As Table 4 shows, frames give no gain in the synchronous regime, whereas in the lagged regime learned frames reduce error by 18% and fixed random frames by only 3%, as predicted by the lag model of Appendix H.1.

Regime	Identity (shared space)	Fixed random frames	Learned frames
Synchronous ($\tau_j = 0$)	1.011±0.039	1.008±0.042	1.016±0.064
Lagged ($\tau_j \in [0, 60]$)	1.112±0.045	1.079±0.076	**0.915**±0.077

Table 4: Theory-matched synthetic test (test MSE, mean ± sample SD over 3 seeds).

7 Conclusion

We separated cross-variable aggregation into routing, which decides which sources contribute and how strongly, and value interfaces, which decide how their information is expressed and interpreted. FrameMix realizes interfaces with variable-specific orthogonal frames, leaving routing unchanged. Across routing rules, frame constructions, and parameterizations, the benefit comes from persistent variable-specific value processing rather than added capacity, mostly on the receiver side. Frameformer is strongest on coupled sensor data and weaker on Exchange, where cross-variable information is of little use. Because FrameMix gives each variable persistent parameters, it assumes a fixed channel set; extending it to unseen or reordered channels (Xu et al., 2026), adding long-horizon PEMS baselines, and comparing with FACT and xCPD are next steps.

AI USE STATEMENT

An LLM-based assistant was used for coding, designing and running supplementary experiments (the synthetic lag test, the Householder-count ablation, the cost measurements, and the frame analysis), aggregating results, drafting the informal analysis and proposition in Appendix H.1, formatting LaTeX tables and figures, and editing the manuscript. The authors independently verified all experimental results and mathematical arguments and take full responsibility for the final content.

ETHICS STATEMENT

This work uses only public forecasting benchmarks that contain aggregated measurements and no personal data, and involves no human subjects. Forecasts can be wrong and should not be the sole basis for safety-critical decisions.

REPRODUCIBILITY STATEMENT

The architecture, Householder frame parameterization, and patch-position modulation are specified in Section 3 and Appendix A. The optimizer, schedule, data splits, normalization, seed protocol, hyperparameters, parameter counts, and compute are given in the training and evaluation protocol of Appendix A. All benchmarks are public, and all methods share the same data loaders, splits, and training-set normalization. Code and configurations will be released upon acceptance.

REFERENCES

Md Sadek Hossain Asif, Tanzila Khan, and Md Mosaddek Khan. Temporal sheaf neural networks with dynamic orthogonal transport. *arXiv preprint arXiv:2606.10071*, 2026.

Jacob Bamberger, Federico Barbero, Xiaowen Dong, and Michael Bronstein. Bundle neural network for message diffusion on graphs. In *International Conference on Learning Representations*, volume 2025, pp. 16788–16813, 2025.

Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, Michael Bronstein, Petar Veličković, and Pietro Liò. Sheaf neural networks with connection laplacians. In *Topological, Algebraic and Geometric Learning Workshops 2022*, pp. 28–36. PMLR, 2022a.

Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, and Pietro Lio. Sheaf attention networks. In *NeurIPS 2022 workshop on symmetry and geometry in neural representations*, pp. 18527–18541, 2022b.

Alessio Borgi, Fabrizio Silvestri, and Pietro Liò. Polynomial neural sheaf diffusion: A spectral filtering approach on cellular sheaves. *arXiv preprint arXiv:2512.00242*, 2025.

Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. Spectral temporal graph neural network for multivariate time-series forecasting. volume 33, pp. 17766–17778, 2020.

Jialin Chen, Jan Eric Lenssen, Aosong Feng, Weihua Hu, Matthias Fey, Leandros Tassiulas, Jure Leskovec, and Rex Ying. From similarity to superiority: Channel clustering for time series forecasting. volume 37, pp. 130635–130663, 2024.

Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=wbpxTuXgm0.

Lu Han, Xu-Yang Chen, Han-Jia Ye, and De-Chuan Zhan. Softs: Efficient multivariate time series forecasting with series-core fusion. volume 37, pp. 64145–64175, 2024a.

Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):7129–7142, 2024b.

Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform. In *International conference on machine learning*, pp. 1969–1978. PMLR, 2018.

Jinkwan Jang, Hyungjin Park, Jinmyeong Choi, and Taesup Kim. Towards robust real-world multivariate time series forecasting: a unified framework for dependency, asynchrony, and missingness. In *International Conference on Learning Representations*, volume 2026, pp. 153495–153528, 2026.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International conference on machine learning*, pp. 3794–3803. PMLR, 2019.

Dongyuan Li, Shun Zheng, Chang Xu, Jiang Bian, and Renhe Jiang. Routing channel-patch dependencies in time series forecasting with graph spectral decomposition. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=uIPAuyno4Z.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International conference on learning representations*, volume 2024, pp. 11116–11140, 2024.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-xl: Long-context transformers for unified time series forecasting. In *International Conference on Learning Representations*, volume 2025, pp. 83982–84006, 2025.

Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *International Conference on Machine Learning*, pp. 2401–2409. PMLR, 2017.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Amit Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, 30(1):20–36, 2011.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Haoxin Wang, Yipeng Mo, Kunlan Xiang, Nan Yin, Honghe Dai, Bixiong Li, Songhai Fan, and Site Mo. Csformer: Combining channel independence and mixing for robust multivariate time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 21090–21098, 2025.

Huiqiang Wang, Jieming Shi, and Li Qing. Fact: Fine-grained across-variable convolution for multivariate time series forecasting. In *The Fourteenth International Conference on Learning Representations*.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. In *International conference on learning representations*, volume 2024, pp. 38626–38652, 2024a.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. volume 37, pp. 469–498, 2024b.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. volume 34, pp. 22419–22430, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. 2019.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 753–763, 2020.

Jiyuan Xu, Wenyu Zhang, Xin Jing, Jiahao Nie, Shuai Chen, and Shuai Zhang. Cpiri: Channel permutation-invariant relational interaction for multivariate time series forecasting. In *International Conference on Learning Representations*, volume 2026, pp. 145879–145900, 2026.

Zhijian Xu, Ailing Zeng, and Qiang Xu. Fits: Modeling time series with $10k$ parameters. In *International Conference on Learning Representations*, volume 2024, pp. 26295–26318, 2024.

Kun Yi, Qi Zhang, Wei Fan, Hui He, Liang Hu, Pengyang Wang, Ning An, Longbing Cao, and Zhendong Niu. Fouriergnn: Rethinking multivariate time series forecasting from a pure graph perspective. volume 36, pp. 69638–69660, 2023.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.

Lifan Zhao and Yanyan Shen. Rethinking channel dependence for multivariate time series forecasting: Learning from leading indicators. In *International Conference on Learning Representations*, volume 2024, pp. 41433–41451, 2024.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pp. 27268–27286. PMLR, 2022.

Tian Zhou, Qingsong Wen, Jinyang Gao, Bolin Ding, Rong Jin, et al. Card: Channel aligned robust blend transformer for time series forecasting. In *International conference on learning representations*, volume 2024, pp. 11245–11283, 2024.

A Architecture and Experimental Details

A.1 FrameMix Architecture

Patch encoder and decoder. The length-L input for each variable is divided into $P = L/s$ non-overlapping patches of length s and projected into $\mathbb{R}^d$ by the shared patch embedding layer with positional embedding. After the blocks, each variable i has a representation $\mathbf{Z}_i \in \mathbb{R}^{P\times d}$; a single shared forecasting head flattens $\mathbf{Z}_i$ and maps $\mathbb{R}^{Pd} \to \mathbb{R}^H$, producing the H-step forecast. The horizon H is set independently of L: across the multi-horizon study, only the output dimension of

the forecasting head changes, while the rest of the architecture remains unchanged. We use $L = 96$ throughout.

Two-stage transformer block. Each of the $n = 2$ blocks contains two sub-layers: (i) FrameMix across variables, applied independently at each patch position, followed by (ii) standard multi-head self-attention across patch positions within each variable. Each sub-layer uses a residual connection and layer normalization, followed by a shared feed-forward network.

Patch-position modulation. To allow each variable-specific frame to vary smoothly with patch position without introducing an explicit time warp, we parameterize the Householder vectors as

$$\mathbf{h}_{i,\ell}^{(p)} = \mathbf{h}_{i,\ell}^{\text{base}} + \sigma(\gamma_i)\ \text{pos}(p)\,\mathbf{h}_{i,\ell}^{\text{dyn}}, \quad \mathbf{U}_i^{(p)} = \prod_{\ell=1}^{k} \mathcal{H}\Big(\mathbf{h}_{i,\ell}^{(p)}\Big),$$

where $\text{pos}(p) \in [-1, 1]$ is a normalized scalar encoding of patch position p, and γ_i is a learnable per-variable gate initialized to -3. Consequently, the dynamic component is largely suppressed at initialization and is learned during training.

A static variant, obtained by removing the modulation term, matches the patch-modulated variant within seed noise on PEMSD8, PEMSD4, Electricity, and ETTh1, while modulation improves Solar (.180 vs. .202). See Appendix G.3 for the full comparison.

Householder frame construction. Each variable-specific frame $\mathbf{U}_i \in \mathbb{R}^{d\times d}$ is parameterized as a product of k Householder reflections:

$$\mathbf{U}_i = \prod_{\ell=1}^{k} \mathcal{H}(\mathbf{h}_{i,\ell}), \qquad \mathcal{H}(\mathbf{h}) = \mathbf{I} - \frac{2\mathbf{h}\mathbf{h}^\top}{\mathbf{h}^\top\mathbf{h}}.$$

Each reflection is orthogonal, so $\mathbf{U}_i$ is orthogonal for any $\mathbf{h}_{i,\ell}$. We use $k = 4$ reflections, requiring $O(kd)$ learnable parameters per variable. The random-frame controls freeze randomly initialized Householder vectors $\mathbf{h}_{i,\ell}$. A product of $k = 4$ Gaussian Householder reflections acts on at most a k-dimensional subspace and should not be conflated with a Haar-uniform orthogonal draw.

Temporal carry and why it is disabled. FrameMix admits an auxiliary temporal carry path that reuses each variable's frames to transport information across adjacent patch positions:

$$\widetilde{\mathbf{v}}_i^{(p)} = \mathbf{v}_i^{(p)} + \Big(\mathbf{U}_i^{(p)}\Big)^\top \mathbf{U}_i^{(p-1)}\mathbf{v}_i^{(p-1)}.$$

This path is *disabled* in all reported results. If enabled, each variable retains a stable frame association outside the cross-variable aggregation operator; consequently, shuffling the frame assignments within that operator would leave the carry path unchanged, confounding the intended intervention. Disabling the carry ensures that the cross-variable FrameMix operator is the sole consumer of the frame parameters, so every reported result corresponds to a single, coherent architecture. Diagnostic tables are obtained from independent training runs and reproduce the main-table Frameformer within 0.005 absolute MSE (the largest difference occurs on Weather).

A.2 Training and Evaluation Protocol

Optimizer and schedule. All models are trained with Adam (Kingma & Ba, 2014) ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$), learning rate 10^{-3}, no warm-up, for 15 epochs with early stopping on validation MSE (patience 5). MSE is the training and evaluation loss throughout.

Data splits and normalization. We follow the standard 7/1/2 train/val/test splits for ETT datasets, and the 6/2/2 splits used by iTransformer for Solar, PEMSD8/4, Weather, and Electricity. All inputs are normalized per-variable to zero mean and unit variance using training-set statistics.

Seed protocol. Unless stated otherwise, results are means over three seeds; the transplantation analyses in Appendix D use five independently trained models. Paired statistical tests (Wilcoxon, paired-t) treat each dataset as one observation.

Model-specific hyperparameters. External baselines use paper-recommended configurations under common loaders. Frameformer uses latent dimension $d = 128$, $k = 4$ Householder reflections per frame, $n = 2$ blocks, patch length $s = 16$, and horizon $H = 96$ unless stated.

Parameter counts and compute. Frameformer at PEMSD4 ($N = 307$, $d = 128$, $k = 4$): $\approx$ 0.79M parameters. Crossformer at PEMSD4: $\approx$ 9.3M parameters (Appendix H.4). All experiments were run on a single NVIDIA A100 40 GB GPU; training one Frameformer run at $H = 96$ takes approximately 8–12 minutes.

Minimal-operator interface protocol. The transplant, readaptation, random-frame, component-hierarchy, and relation-operator experiments use a minimal faithful operator: a shared linear encoder, per-variable orthogonal Householder frames, a per-instance attention router, and a shared linear decoder. Its value-path output is verified against the corresponding full PyTorch (Paszke et al., 2019) operator to numerical tolerance (10^{-6}). Five independent seeds are used for directed transplantation. On PEMSD4 and PEMSD8, the variable count is capped at $N = 25$ for tractable all-pairs sweeps; ETTh1, Weather, and Exchange use their audited variable sets. This protocol isolates frame and component compatibility and is distinct from the full-forecaster evaluations.

A.3 Implementation Details

Attention and value-path implementation. At each patch position p, the cross-variable FrameMix operator changes only the *value* aggregation path of the cross-variable attention:

$$\widehat{\mathbf{v}}_i^{(p)} = \left(\mathbf{U}_i^{(p)}\right)^\top \sum_{j=1}^{N} A_{ij}^{(p)} \mathbf{U}_j^{(p)} \mathbf{v}_j^{(p)},$$

where $A_{ij}^{(p)} = \mathrm{softmax}_j\big(\langle \tilde{\mathbf{v}}_i^{(p)}, \tilde{\mathbf{v}}_j^{(p)} \rangle / \sqrt{d}\big)$ is a parameter-free scaled dot-product weight over the layer-normalized tokens $\tilde{\mathbf{v}}$; Frameformer has no query, key, or value projections in this layer. In a host with learned query and key projections, such as iTransformer (Appendix G.2), FrameMix is a drop-in replacement for the value aggregation path and leaves the query–key computation and attention weights unchanged.

Shuffle controls. We use two shuffles that intervene on different components of the cross-variable mixer. To disrupt routing while preserving the distribution of routing weights within each receiver row, we sample a uniform permutation π over the variable indices and replace

$$A_{ij}^{(p)} \quad \text{with} \quad A_{i,\pi(j)}^{(p)}.$$

To disrupt variable–frame ownership with routing weights unchanged, we replace

$$\mathbf{U}_i^{(p)} \quad \text{with} \quad \mathbf{U}_{\pi(i)}^{(p)}.$$

During training, a new permutation is sampled at every forward pass. During evaluation, a new permutation is sampled for each batch, and the reported test-time shuffle result is averaged over five permutation draws per seed.

Identity-recovery numerical check. Setting all source and receiver interfaces to the identity, $\mathbf{S}_j = \mathbf{T}_i = \mathbf{I}$ in Eq. (1) for every i, j, in the iTransformer drop-in experiment exactly recovers the original iTransformer output, with zero numerical difference over 100 randomly generated inputs.

Reproducibility. All code and configurations will be released upon acceptance. Random seeds are fixed via `torch.manual_seed` and `numpy.random.seed` at the start of each run.

B Routing-Shuffle Controls

B.1 Synthetic Sensitivity Validation

To verify that the shuffle control detects genuine structure when present, we construct two synthetic data-generating processes (DGPs) with installed cross-variable dependencies and confirm that the control produces the expected signature.

Clustered DGP. Variables are partitioned into clusters of size 4. Within each cluster, all series share a latent factor plus independent noise. An iTransformer trained on this data exploits the cluster structure, incurring a test-time shuffle penalty of $+331.7\%$. Under train-time shuffle, the penalty remains substantial ($+287.3\%$), confirming that the learned routing structure is both committed to and necessary in this DGP, unlike the real-data setting, where train-time shuffle recovers accuracy.

Dataset	learned	frozen rand.	identity	**U**-shuf
ETTh1	.455±.004	.457±.003	.462±.003	.515±.018
Weather	.149±.002	.150±.002	.169±.003	.226±.011
Solar	.180±.005	.202±.009	.192±.006	.228±.014
PEMSD8	.178±.002	.211±.005	.257±.007	.517±.032
PEMSD4	.111±.001	.153±.004	.212±.006	.650±.041
Electricity	.140±.003	.143±.003	.153±.004	.300±.019

Table 5: Frame presence: test MSE ($h = 96$, 3-seed mean ± std). Independent diagnostic reruns, so values differ slightly from Tables 1 and 3.

Donor frames	Model 0	Model 1	Model 2	Model 3	Model 4
$\mathcal{U}^{(0)}$	.18	1.07	.87	.82	.92
$\mathcal{U}^{(1)}$	.96	.18	.88	.84	1.04
$\mathcal{U}^{(2)}$	.98	.91	.19	.93	.89
$\mathcal{U}^{(3)}$	1.07	.81	1.09	.18	.77
$\mathcal{U}^{(4)}$	.87	1.20	.85	1.01	.19

Table 6: Directed frame transplantation on PEMSD4. Entry (a, b) is the test MSE of recipient model b after replacing its frame set with donor model a's frame set $\mathcal{U}^{(a)}$. The diagonal mean is .184, while the off-diagonal mean is .939 ($5.11\times$ higher).

Directed-coupling DGP. Variable j causally influences variable i with a one-step lag, and only these directed dependencies are informative. The iTransformer exploits this structure (test-time shuffle $+29.4\%$); train-time shuffle degrades performance similarly ($+24.1\%$), again confirming that the installed structure is necessary. Across both DGPs, the control clearly separates the genuine-structure regime from the substitutable regime observed on real datasets, validating its sensitivity.

B.2 Within-Host Topology Checks

To confirm that the topology conclusion also holds within the FrameMix architecture, we perform routing-weight shuffle controls by permuting the routing matrix $\mathbf{A}^{(p)}$ while evaluating the shared-frame ($\mathbf{U}_i^{(p)} = \mathbf{I}$) and variable-specific-frame (FrameMix) variants separately.

In both cases, train-time routing-weight shuffling recovers within 2% of normal performance on all six datasets, whereas test-time routing-weight shuffling substantially degrades performance ($+5\%$–65%, depending on the dataset).

C Full Frame Controls

C.1 Frame Presence

Table 5 extends the frame-presence comparison of Table 1 with standard deviations and the test-time frame-assignment shuffle condition. The learned–identity and learned–frozen-random gaps are consistent in direction across all three seeds on every dataset. The frozen-random–identity comparison is directionally consistent on five of the six datasets; Solar reverses on one seed, consistent with the observation in the main paper.

C.2 Frame Assignment

T1/T2 details. For T1 (test-time frame-assignment shuffle), we sample a fresh permutation π over the N variables for each evaluation batch and replace

$$\mathbf{U}_i^{(p)} \leftarrow \mathbf{U}_{\pi(i)}^{(p)} \qquad \text{for all } i \in \{1, \ldots, N\} \text{ and } p \in \{1, \ldots, P\}.$$

The reported result is averaged over five independent permutation draws per seed.

Dataset	attention **Q/K**	frames	decoder head
PEMSD4	$1.13\times$	$5.11\times$	$6.91\times$
Exchange	—	$0.78\times$	$3.50\times$

Table 7: Component-transplant hierarchy: mean off-diagonal/diagonal MSE ratio when one component is transplanted across five seeds ($1.0\times$ = no penalty; Q/K not run on Exchange). Q/K denotes the learned router projections of the minimal operator (Appendix A.3); Frameformer itself has none.

For T2 (train-time frame-assignment shuffle), a fresh permutation π is sampled at every training step and applied in the same manner. The model is trained to convergence and then evaluated using its own learned variable-specific frames, without frame reassignment at test time.

Reshuffle consistency check. For each T2 model, we apply a fresh test-time frame-assignment shuffle, denoted T2-reshuf, and verify that its test MSE differs from $\mathcal{E}_{\mathrm{T2}}$ by at most 0.001. This agreement confirms that the T2 model has lost its dependence on a specific variable–frame assignment rather than memorizing the distribution of training-time permutations.

D Co-Adapted Interface Experiments

All experiments in this section use the minimal faithful operator described in Appendix A.2. The full Frameformer establishes predictive performance and the importance of persistent variable–frame assignment, whereas the experiments below isolate the compatibility between variable-specific frames and neighboring model components across independently trained runs.

D.1 Five-Seed Directed Transplantation on PEMSD4

Table 6 reports the directed transplantation matrix. Let

$$\mathcal{U}^{(a)} = \left\{ \mathbf{U}_i^{(p,a)} \right\}_{i=1,p=1}^{N,P}$$

denote the complete set of patch-dependent variable frames learned by donor model a. Entry (a, b) reports the test MSE of recipient model b after replacing its frame set $\mathcal{U}^{(b)}$ with donor frame set $\mathcal{U}^{(a)}$, while keeping all other parameters of recipient model b fixed. All models use the same dataset, architecture, and training protocol but are trained with independent random seeds.

All 20 directed off-diagonal transplantations increase test error. The matrix is not symmetric because the non-frame parameters of the recipient models differ across independently trained runs.

D.2 Foreign-Frame Readaptation

Two normally trained PEMSD4 models obtain test MSEs of .1798 (model A) and .1802 (model B). Let $\mathcal{U}^{(A)}$ and $\mathcal{U}^{(B)}$ denote the complete sets of variable-specific frames learned by models A and B, respectively. Replacing only model B's frame set $\mathcal{U}^{(B)}$ with $\mathcal{U}^{(A)}$, while keeping all other parameters of model B fixed, increases the test MSE to .9763. Randomly permuting the variable ownership of model B's own frames produces an MSE of .7962.

We then fix the transplanted donor frame set $\mathcal{U}^{(A)}$ and retrain all remaining parameters of model B, obtaining an MSE of .1789. Thus, the same donor frames that fail under post-hoc transplantation support a fully accurate model once the surrounding modules are allowed to readapt.

D.3 Identity, Random, and Learned Refit Decomposition

On PEMSD4, the frame-free reference obtains an MSE of .921. Refitting the surrounding model under the fixed identity convention

$$\mathbf{U}_i^{(p)} = \mathbf{I} \qquad \text{for all } i \in \{1, \ldots, N\} \text{ and } p \in \{1, \ldots, P\}$$

Dataset (N)	Variant	Acc. N	Param ratio	Δerr (%)
PEMSD8 (170)	pair map	25	$85\times$	$+2.1$
	basis residual	170	$1.7\times$	$+15.2$
	edge residual	25	$86\times$	$+4.3$
PEMSD4 (307)	pair map	25	$154\times$	$+1.9$
	basis residual	307	$2.2\times$	$+5.2$
	edge residual	25	$154\times$	-3.3
Electricity (321)	pair map	25	$161\times$	$+1.1$
	basis residual	321	$2.3\times$	$+11.8$
	edge residual	25	$162\times$	$+1.3$
Weather (21)	pair map	21	$10.5\times$	$+1.5$
	basis residual	21	$1.2\times$	-1.9
	edge residual	21	$11.5\times$	$+0.9$
Exchange (8)	pair map	8	$4.0\times$	$+4.4$
	basis residual	8	$1.3\times$	$+7.3$
	edge residual	8	$5.0\times$	$+17.6$
ETTh1 (7)	pair map	7	$3.5\times$	-0.7
	basis residual	7	$1.3\times$	-2.8
	edge residual	7	$4.5\times$	-0.5

Table 8: Relation-operator relaxations behind Figure 6. Δerr = test-MSE change vs. Frameformer (lower is better); the quadratic variants use a capped $N{=}25$ for accuracy but full-N parameter ratios (so "53M vs. 329k" is a full-N figure).

Dataset	sync residual	FM–data residual
Exchange	.337	.637
Electricity	.431	.723
ETTh1	.481	.827
Weather	.612	.968
PEMSD4	.737	.951

Table 9: Static transport audit (lower = closer FrameMix–data match or greater synchronizability).

reduces the MSE to .395, recovering 71% of the absolute gap between the frame-free reference and learned FrameMix. Using fixed random frame sets, four independent draws obtain $.277 \pm .002$, corresponding to an 87% recovery, while learned frames reach .184. This comparison separates the contributions of readaptation, a persistent orthogonal scaffold, and learned frame selection.

On PEMSD8, learned frames do not improve over fixed random frames beyond seed variation: the learned-frame model obtains $.300 \pm .027$, the fixed-random-frame model obtains $.296 \pm .010$, and a learned model trained for 20 epochs obtains .301. Thus, the benefit of learned frame selection is dataset-dependent, whereas the usefulness of a persistent variable-specific frame assignment replicates across these settings.

D.4 Component-Transplant Hierarchy

Repeating the five-seed transplantation analysis for individual model components gives the mean ratio between off-diagonal and matched test MSE summarized in Table 7.

The PEMSD4 result shows that cross-seed transplantation failure is not a generic property of all learned parameters. The Exchange result further shows that decoder non-transferability can persist even where compatibility between variable-specific frames does not predict forecasting accuracy.

D.5 Relation-Operator Audit

We compare FrameMix with three pair-specific alternatives. The pair map replaces the node factorization by a learned ordered-pair operator. The basis residual inserts

$$\mathbf{E}_{ij} = \mathbf{I} + \sum_b c_{ijb}\mathbf{B}_b$$

Dataset	shared	FrameMix	difference	norm ratio
ETTh1	.53	.43	-.10	.76
Weather	.55	.39	-.16	.85
Electricity	.67	.38	-.29	.60
Exchange	.92	.48	-.45	.96
PEMSD4	.87	.25	-.63	.41

Table 10: Cosine compatibility of the aggregate with the target representation; FrameMix lowers it.

Dataset	no-mix $\mathbf{v}$	shared	FrameMix
PEMSD4	6.30	4.19	30.77
Exchange	4.81	4.85	17.39
ETTh1	26.22	26.58	42.82
Weather	9.82	12.74	36.38

Table 11: Effective rank of latent values: no mixing, shared-map mixing, and FrameMix.

between the source and receiver maps, and the edge residual gives every ordered pair an independent low-rank residual. The residual coefficients are initialized at zero. Unit tests verify that this setting reproduces FrameMix exactly (maximum forward difference 0.00) and that all relation parameters receive gradients.

The results are reported in Table 8. The learned residuals are not dormant: on PEMSD4 and Weather, mean $\|\mathbf{E}_{ij} - \mathbf{I}\|_F$ lies between 2.7 and 4.3 and the fraction of near-identity pairs is zero. The added pair-specific freedom is therefore used, yet the basis residual degrades PEMSD4 ($+5.2\%$) and the edge residual improves it (-3.3%) only at the capped $N = 25$, where its full-N version would need $154\times$ the interface parameters; the lack of a reliable gain is not an implementation artifact.

E Static-Geometry Audit

The following analyses test, rather than assume, that learned frames recover an intrinsic latent geometry. They are retained as a negative mechanism audit because they delimit the interpretation of the learned maps. The PEMS setting in this audit is PEMSD4.

E.1 Pairwise Transport and Synchronization

For each variable pair, we estimate an independent orthogonal Procrustes map from latent representations and compare it with the learned relative map $\mathbf{R}_{ij} = \mathbf{U}_i^\top \mathbf{U}_j$. We also measure the residual of their best node-factorized synchronization.

Table 9 summarizes both quantities. Across all datasets, the synchronization residual remains substantial, and the learned relative maps also exhibit large FrameMix–data residuals. Thus, the learned interfaces provide little evidence for a shared latent geometry that is consistent across variables.

Table 10 further compares the cosine compatibility between the aggregated representation and the target representation under shared-space mixing and FrameMix. Surprisingly, FrameMix consistently *reduces* this compatibility across all datasets. Thus, the gains from FrameMix cannot be explained by making the aggregated representation more closely aligned with the receiver representation in the original latent space.

Attention weighting does not rescue the account: uniform/attention-weighted residuals are .827/.756 (ETTh1), .723/.706 (Electricity), .968/.927 (Weather), .637/.637 (Exchange), and .951/.952 (PEMSD4). Neither PEMSD4 nor Exchange shows preferential fit on high-attention edges.

E.2 Effective Rank and the No-Mix Baseline

Table 11 compares the effective rank of the latent values before cross-variable aggregation (no-mix), after shared-space aggregation, and after FrameMix. Across all datasets, FrameMix increases the

Dataset	Frameformer	iTransf.	DLinear	PatchTST	TimeXer	CARD	CSformer
Horizon $H = 192$							
ETTh1	**.511**	.524	.516	.539	.530	.655	.512
ETTh2	.318	**.300**	.324	.303	.310	.401	.305
Exchange	.233	.228	**.206**	.247	.236	.344	.247
Weather	**.206**	.226	.239	.224	.209	.211	.219
Solar	**.203**	.237	.312	.242	.230	.225	.227
Electricity	**.158**	.170	.195	.179	.159	—	—
Horizon $H = 336$							
ETTh1	**.555**	.584	.561	.580	.569	.732	.590
ETTh2	**.350**	.360	.398	.363	.354	.385	.357
Exchange	.504	**.408**	.438	.455	.419	.608	.417
Weather	.273	.279	.291	.275	**.264**	.271	.267
Solar	**.218**	.249	.346	.270	.244	.243	.264
Electricity	.181	.184	.208	.195	**.177**	—	—
Horizon $H = 720$							
ETTh1	.678	.695	**.651**	.686	.704	.843	.732
ETTh2	**.456**	.459	.592	.470	.487	.584	.461
Exchange	1.489	1.058	1.538	1.172	1.149	**.968**	1.103
Weather	.349	.357	.355	.353	**.344**	.350	.345
Solar	**.205**	.250	.349	.266	.247	.236	.257
Electricity	.221	.222	.243	.234	**.216**	—	—

Table 12: Multi-horizon test MSE (3-seed mean), $H \in \{192, 336, 720\}$. Bold indicates the lowest MSE in each row. Crossformer is omitted because it diverged on ETTh2 and Exchange. CARD and CSformer were not evaluated on Electricity (—).

effective rank far more than shared-space aggregation does. This confirms that assigning variable-specific orthogonal frames diversifies the latent value representations. Rank expansion alone need not explain the predictive gains, since frame reassignment (Table 5) degrades accuracy even though it leaves the frames themselves unchanged.

F Multi-Horizon Evaluation

F.1 Full Multi-Horizon Benchmark Results

Table 12 reports the full $H \in \{192, 336, 720\}$ benchmark. Frameformer stays competitive across horizons and is dominant on Solar; TimeXer becomes competitive on Weather and Electricity at long range, and Frameformer stays weak on Exchange.

The main exception is Exchange at $H = 720$, where Frameformer (1.489) falls behind iTransformer (1.058) and CARD (.968). This is the regime where cross-variable coupling is weakest; the variable-specific interface does not help here and appears to fit long-range noise, though we do not isolate the precise cause.

F.2 PEMS Horizon Curves

We report Frameformer at all four horizons on the PEMS datasets; the baselines were not rerun at long horizons under our compute budget. Test MSE grows gradually with horizon—PEMSD8 .178/.257/.316/.359 and PEMSD4 .111/.126/.133/.158 at $H = 96/192/336/720$—without the sharp long-horizon degradation seen on Exchange. The interface thus remains useful precisely where cross-variable coupling is strong.

Dataset	vanilla	FM-learned	FM-affine	affine vs. $\mathbf{v}$
PEMSD8	$.361 \pm .001$	$.349 \pm .004$	$.309 \pm .012$	-14.3%
PEMSD4	$.309 \pm .002$	$.186 \pm .004$	$.164 \pm .005$	-47.1%
Weather	$.175 \pm .001$	$.163 \pm .002$	$.162 \pm .001$	-7.5%
ETTh1	$.470 \pm .001$	$.475 \pm .008$	$.470 \pm .001$	-0.1%

Table 13: Drop-in transfer to iTransformer ($h = 96$, 3-seed mean $\pm$ std). Both per-variable maps improve vanilla on PEMS and Weather but not ETTh1; the orthogonal margin inverts here.

G Additional Parameterization Controls

G.1 Affine Interface Control

To verify that the frame gain does not merely reflect the Householder parameterization, we replace the orthogonal source and receiver frames with independent per-variable affine maps $\mathbf{S}_j(\mathbf{v}) = \mathbf{s}_j^{\mathrm{src}} \odot \mathbf{v} + \mathbf{b}_j^{\mathrm{src}}$ and $\mathbf{T}_i(\mathbf{z}) = \mathbf{s}_i^{\mathrm{tgt}} \odot \mathbf{z} + \mathbf{b}_i^{\mathrm{tgt}}$ inside FrameMix, giving

$$\widehat{\mathbf{v}}_i = \mathbf{T}_i \left(\sum_{j=1}^{N} A_{ij}\, \mathbf{S}_j(\mathbf{v}_j) \right).$$

The two maps use $4d$ parameters per variable. Since the deployed Frameformer uses patch-modulated frames ($\approx 2kd$ frame parameters per variable at $k = 4$), the affine map is a *lower-parameter*, non-orthogonal control rather than an exactly parameter-matched one; it remains far below one dense operator per ordered pair. That a lightweight affine map matches the orthogonal frame strengthens the conclusion that the load-bearing factor is a persistent per-variable transformation rather than the orthogonal form specifically.

Results (Table 2): the affine control recovers 91%–94% of the identity gap on the PEMS datasets and 83% on Weather, but only 38% on Solar. It slightly outperforms the orthogonal frame on Electricity and is outperformed on the other four datasets. The orthogonal–affine margin is small overall and not statistically significant (paired Wilcoxon over the five datasets in Table 2, two-sided $p = 0.125$); the ordering further reverses in the iTransformer drop-in below. In the iTransformer drop-in experiment (Appendix G.2), the affine correction improves PEMSD4 by 47.1%. Thus the robust conclusion is not that orthogonality is universally optimal, but that per-variable transformation around the mixer is the load-bearing intervention.

G.2 Drop-In Transfer to iTransformer

To confirm the correction is not load-bearing only inside an architecture designed around it, we insert it directly into iTransformer's variate attention, replacing its value path with

$$\widehat{\mathbf{v}}_i = \mathbf{T}_i \left(\sum_{j=1}^{N} A_{ij}\, \mathbf{S}_j(\mathbf{v}_j) \right)$$

at full d-dimensional width, using the same independent source and receiver maps $\mathbf{S}_j, \mathbf{T}_i$ as above; setting $\mathbf{S}_j = \mathbf{T}_i = \mathbf{I}$ recovers the original iTransformer numerically (0 output difference, verified). The resulting performance is summarized in Table 13.

The value-interface intervention transfers most strongly where the frame-ablation gap was largest: the affine variant improves iTransformer by 47.1% on PEMSD4 and 14.3% on PEMSD8; ETTh1 remains within 1%, the weak-frame-gain regime. The affine variant outperforms the orthogonal one inside iTransformer, confirming that the transferable claim concerns variable-specific maps around the mixer rather than the Householder form.

Dataset	Static	Modulated
ETTh1	.455	.455
Weather	.150	.149
Solar	.202	**.180**
PEMSD8	.179	.178
PEMSD4	.112	.111
Electricity	.141	.140

Table 14: Static vs. patch-modulated FrameMix (test MSE, $h = 96$, 3-seed mean).

G.3 Static vs. Patch-Modulated Frames

Table 14 summarizes the comparison between static and patch-modulated FrameMix. Solar is the only dataset with a material gap between the static and patch-modulated learned frames. Because a fixed Hadamard frame also performs well on Solar (Table 1), this comparison does not isolate patch dependence as the underlying mechanism; we therefore report it as an implementation-specific exception rather than assigning it a causal interpretation. On all other datasets the static and modulated variants agree within seed noise. The main paper uses the modulated variant throughout.

H When and Why Value Interfaces Help

H.1 An Informal Characterization

We consider one receiver i and linear value processing. Suppose the value that source j should contribute to receiver i is $\mathbf{T}_{ij}^{\star}\mathbf{v}_j$ for some ideal transfer map $\mathbf{T}_{ij}^{\star} \in \mathbb{R}^{d\times d}$, so that the ideal aggregate is $\sum_j A_{ij}\mathbf{T}_{ij}^{\star}\mathbf{v}_j$.

The statements below are for this linear aggregation setting; they are not necessity results for a full nonlinear forecaster.

Shared-space mixing. With $\mathbf{S}_j = \mathbf{T}_i = \mathbf{I}$, the mixer outputs $\sum_j A_{ij}\mathbf{v}_j$, and every later module that is shared across variables (the temporal encoder and the forecasting head of Frameformer) sees only this sum. A linear map $\mathbf{G}$ applied after aggregation gives $\sum_j A_{ij}\mathbf{G}\mathbf{v}_j$, so it reproduces the ideal aggregate for all inputs only if $\mathbf{T}_{ij}^{\star} = \mathbf{G}$ for every source with $A_{ij} > 0$. Within this linear setting, shared-space mixing is therefore sufficient only when the ideal transfer is common to all pairs, for example when variables are synchronous copies of one another or when cross-variable information is irrelevant, as on Exchange. It is insufficient when the ideal transfer depends on the pair.

Proposition 1 (linear setting). Let $A_{ij} > 0$ for $j \in \mathcal{N}_i$. (a) A receiver-shared map $\mathbf{G}$ applied after aggregation satisfies $\mathbf{G}\sum_j A_{ij}\mathbf{v}_j = \sum_j A_{ij}\mathbf{T}_{ij}^{\star}\mathbf{v}_j$ for all $\{\mathbf{v}_j\}$ if and only if $\mathbf{T}_{ij}^{\star} = \mathbf{G}$ for every $j \in \mathcal{N}_i$. (b) A family of orthogonal maps $\{\mathbf{R}_{ij}\}$ on a connected graph can be written as $\mathbf{R}_{ij} = \mathbf{U}_i^{\top}\mathbf{U}_j$ with orthogonal $\mathbf{U}_i$ if and only if $\mathbf{R}_{ii} = \mathbf{I}$ and $\mathbf{R}_{ik}\mathbf{R}_{kj} = \mathbf{R}_{ij}$ for all i, j, k; the frames are then unique up to a common left factor, $\mathbf{U}_i \mapsto \mathbf{Q}\mathbf{U}_i$. *Proof.* (a) Setting all but one $\mathbf{v}_j$ to zero gives $A_{ij}\mathbf{G}\mathbf{v}_j = A_{ij}\mathbf{T}_{ij}^{\star}\mathbf{v}_j$ for all $\mathbf{v}_j$; the converse is immediate. (b) Necessity follows from $\mathbf{U}_i^{\top}\mathbf{U}_i = \mathbf{I}$ and $\mathbf{U}_i^{\top}\mathbf{U}_k\mathbf{U}_k^{\top}\mathbf{U}_j = \mathbf{U}_i^{\top}\mathbf{U}_j$. For sufficiency, fix a variable r and set $\mathbf{U}_j = \mathbf{R}_{rj}$; then $\mathbf{U}_i^{\top}\mathbf{U}_j = \mathbf{R}_{ir}\mathbf{R}_{rj} = \mathbf{R}_{ij}$, using $\mathbf{R}_{ri}^{\top} = \mathbf{R}_{ir}$, which follows from $\mathbf{R}_{ir}\mathbf{R}_{ri} = \mathbf{R}_{ii} = \mathbf{I}$. If $\mathbf{U}_i^{\top}\mathbf{U}_j = \mathbf{V}_i^{\top}\mathbf{V}_j$ for all pairs, then $\mathbf{Q} = \mathbf{V}_r\mathbf{U}_r^{\top}$ satisfies $\mathbf{V}_j = \mathbf{Q}\mathbf{U}_j$. □

Part (b) states both the expressivity limit of FrameMix, which cannot represent inconsistent or asymmetric pairwise transfers, and its identifiability: only the relative maps $\mathbf{R}_{ij}$ are meaningful, so the analyses in Appendix H.6 focus on $\mathbf{R}_{ij}$.

Interfaces. A receiver map realizes $\mathbf{T}_{ij}^{\star} = \mathbf{T}_i$, a source map realizes $\mathbf{T}_{ij}^{\star} = \mathbf{S}_j$, and the two together realize every node-factorized transfer $\mathbf{T}_{ij}^{\star} = \mathbf{T}_i\mathbf{S}_j$ with N maps per side instead of N^2 pair maps.

A lag model in which orthogonal frames are exact. Let each variable be a lagged copy of a common driver, $x_j(t) = s(t - \tau_j)$, and let the latent value contain, for a frequency ω, the phase coordinates $(\cos, \sin)(\omega t - \omega\tau_j)$. Expressing variable j's coordinates in variable i's phase is a rotation by $\omega(\tau_j - \tau_i)$. Writing $\mathbf{U}_j = \mathrm{Rot}(\omega\tau_j)$ gives $\mathbf{T}_{ij}^{\star} = \mathbf{U}_i^{\top}\mathbf{U}_j$, which is exactly the FrameMix

transfer in Eq. (3): it is orthogonal, node-factorized, and consistent ($\mathbf{T}^\star_{ik}\mathbf{T}^\star_{kj} = \mathbf{T}^\star_{ij}$). Shared-space mixing, in contrast, averages out-of-phase coordinates before any later module can act: two equally weighted components with phase difference Δ average to amplitude $|\cos(\Delta/2)|$, so the information is attenuated, and at $\Delta = \pi$ cancelled, inside the mixer. A shared embedding does not remove this mismatch because it applies the same map to every variable. This is the representation mismatch that remains after shared embedding: the embedding makes values comparable, but it does not place differently lagged or phased variables in a common convention.

Why orthogonal. In the lag model the ideal transfers are rotations. More generally, orthogonal interfaces preserve value norms, so they change how a contribution is represented without changing how strongly it counts; the magnitude stays under the control of the routing weights. This separation is what lets us vary interfaces while holding routing fixed. Recovering node frames from pairwise orthogonal transfers is the synchronization problem over the orthogonal group (Singer, 2011), and the resulting node factorization is the flat-bundle structure used by Bundle Neural Networks (Bamberger et al., 2025). We do not claim a connection to optimal transport; the closest classical formulation is orthogonal Procrustes alignment.

Scope. The lag model is a sufficient case in which interfaces are exact, not a description of what learned frames recover. Appendix E tests this directly and finds that learned relative maps do not match per-pair Procrustes estimates, and Appendix D shows that frames are not portable across runs. The lag model is an existence example that motivates the design; it does not by itself explain the mechanism of the trained model. On real data, the evidence points to persistent variable-specific transformation and co-adaptation of the surrounding modules, with orthogonality as an efficient, norm-preserving instantiation rather than a requirement (Table 2). Appendix H.8 tests the prediction of the lag model directly on synthetic data where the lags are known.

Fixed random frames. A fixed random frame set satisfies the structural part of this account, a persistent convention per variable composed as $\mathbf{U}_i^\top \mathbf{U}_j$, without fitting any lag. The surrounding shared modules can then co-adapt to these conventions, which is consistent with fixed frames recovering much of the gain (Table 1), with transplanted frames failing and then working again after readaptation (Appendix D), and with learned frames adding the largest gains on PEMS.

H.2 Why the Receiver Side Dominates

Section 6 finds that receiver-only interfaces recover most of the gain while source-only interfaces are weak. We offer two complementary explanations. First, the patch embedding is shared, so sources already express their values in a common learned space; in the notation above, the source factor of $\mathbf{T}^\star_{ij} = \mathbf{T}_i\mathbf{S}_j$ is largely absorbed by the shared embedding, and the remaining mismatch is on the receiver side. Second, the receiver map is the only variable-specific parameter between the mixer and the shared temporal encoder and head; it re-expresses the update that is added to the receiver's residual stream and acts as a per-variable adapter for the shared downstream modules. A source-only map instead rotates each contribution into a different convention with no matching inverse, so the mixture reaching the receiver combines inconsistent coordinates. The tied form restores consistency, and its extra gain over the receiver-only form appears where many strongly weighted sources are combined (Solar, PEMSD8, PEMSD4), where source-specific expression matters most.

H.3 Frame Parameterization, Initialization, and Patch Dependence

Each Householder vector is initialized i.i.d. Gaussian with standard deviation 0.1; because a reflection depends only on the direction of $\mathbf{h}$, each initial frame is a random product of k reflections. Frames are not independent per patch position. Each variable has a base and a modulation vector per reflection and one gate, $\mathbf{h}^{(p)}_{i,\ell} = \mathbf{h}^{\text{base}}_{i,\ell} + \sigma(\gamma_i)\,\text{pos}(p)\,\mathbf{h}^{\text{dyn}}_{i,\ell}$ with γ_i initialized to -3, giving $2kd + 1$ frame parameters per variable independent of the number of patches. This lets a variable's convention drift smoothly across the window, a simple form of temporal evolution, while keeping one persistent identity. Appendix H.4 lists the resulting frame parameters: 0.31M of 0.79M parameters on PEMSD4 and 0.02M of 0.50M on Weather. A static variant without modulation matches the modulated one within seed noise except on Solar (Table 14).

H.4 Computational Cost

	Weather ($N = 21$)				PEMSD4 ($N = 307$)				Electricity ($N = 321$)			
Model	Params	Train	Infer.	Mem.	Params	Train	Infer.	Mem.	Params	Train	Infer.	Mem.
DLinear	19k	0.4	0.1	0.0	19k	0.4	0.1	0.1	19k	0.4	0.1	0.1
PatchTST	0.41M	2.2	0.6	0.2	0.41M	21.5	5.2	1.9	0.41M	21.9	5.6	2.0
iTransformer	0.22M	2.0	0.5	0.0	0.22M	2.7	0.8	0.4	0.22M	3.0	0.8	0.4
TimeXer	1.79M	4.4	0.9	0.2	1.86M	50.0	8.1	2.8	1.87M	53.3	8.7	3.0
Crossformer	8.10M	13.6	4.3	1.0	9.27M	115.4	33.7	12.0	9.33M	119.6	35.1	12.5
CARD	0.98M	8.3	2.5	0.6	0.98M	105.6	28.5	7.2	0.98M	111.7	30.4	7.5
CSformer	0.64M	11.1	2.4	0.8	0.31M	96.7	16.3	3.2	0.31M	117.3	19.9	3.3
Frameformer	0.50M (22k)	5.5	1.2	0.2	0.79M (0.31M)	36.8	5.8	3.4	0.80M (0.33M)	37.4	6.1	3.5

Table 15: Computational cost at $h = 96$: trainable parameters (Frameformer frame parameters in parentheses), milliseconds per training step, milliseconds per inference batch, and peak GPU memory (GiB), measured alone on one NVIDIA RTX PRO 6000 GPU. CSformer uses a smaller width on the three large datasets, following its benchmark configuration.

Table 15 reports parameters, time per training step (forward, backward, and optimizer step on one batch), inference time per batch, and peak memory for every model at $h = 96$, measured alone on one NVIDIA RTX PRO 6000 GPU with each model's benchmark configuration (batch 32; 16 for CSformer on the large datasets), after 5 warm-up steps and averaged over 20 steps. Frame composition is never materialized as a $d \times d$ matrix: applying k reflections to N tokens costs $O(Nkd)$ per patch and direction, against $O(N^2 d)$ for the aggregation itself, so the interface adds a small constant factor. The routing weights are a parameter-free scaled dot-product softmax over layer-normalized tokens, so FrameMix adds no query or key parameters.

H.5 Number of Householder Reflections

	$k = 1$	$k = 2$	$k = 4$	$k = 8$	$k = 16$
PEMSD8	0.182±0.002	0.177±0.004	0.175±0.003	0.176±0.002	0.182±0.002
Weather	0.155±0.002	0.152±0.004	0.154±0.005	0.155±0.002	0.151±0.001

Table 16: Sensitivity to the number of Householder reflections per frame (test MSE, $h = 96$, mean $\pm$ sample SD over 3 seeds; all other settings as in the benchmark). The paper uses $k = 4$.

Performance is flat in k over a wide range (Table 16). On Weather all settings lie within seed variation. On PEMSD8, $k = 2$ to 8 are within 1% of each other, while a single reflection ($k = 1$) and $k = 16$ are about 4% worse; one reflection restricts each frame to a rank-one change of the identity, and 16 reflections enlarge the per-variable parameter count and make optimization harder without adding accuracy. The reflections are applied sequentially, so training time grows with k. We use $k = 4$ as a mid-range setting; the results do not depend on tuning it. We did not try Cayley or matrix-exponential parameterizations (Helfrich et al., 2018; Lezcano-Casado & Martınez-Rubio, 2019); the fixed Hadamard and non-orthogonal affine controls (Tables 1 and 2) indicate that the specific orthogonal parameterization is not what carries the gain.

H.6 What Learned Frames Encode

We analyze the learned frames of the three PEMSD8 benchmark-configuration models ($k = 4$, 3 seeds) against their own initializations. Because a product of k reflections acts on at most a k-dimensional subspace, $\mathrm{tr}(\mathbf{R}_{ij})/d$ is close to one by construction, so we compare its spread and rank correlations rather than its level. Road adjacency uses the public PEMS08 sensor graph (295 edges).

Statistic (PEMSD8, $k = 4$, mean $\pm$ SD over 3 seeds)	Initialization	Learned
Overlap with initialization, $\mathrm{tr}(\mathbf{U}_i^{(0)\top}\mathbf{U}_i)/d$	–	0.936±0.001
Overlap of first and last patch frames	0.999±0.000	0.993±0.000
Patch-modulation gate $\sigma(\gamma_i)$, mean	0.047±0.000	0.075±0.001
SD of pairwise alignment $a_{ij} = \mathrm{tr}(\mathbf{R}_{ij})/d$	0.0017±0.0000	0.0063±0.0004
Spearman$(a_{ij}, \lvert\rho_{ij}\rvert)$, training-series correlation	0.05±0.05	0.23±0.05
Mean a_{ij}, road-adjacent pairs	0.882±0.000	0.902±0.001
Mean a_{ij}, pairs ≥ 5 hops apart	0.882±0.000	0.896±0.001
Spearman$(a_{ij}, -\text{hops})$	0.00±0.02	0.06±0.04

Table 17: What learned frames encode on PEMSD8.

Table 17 supports three observations. First, frames move substantially from their random initialization: the learned frame overlaps its initialization about as much as it overlaps the identity, so training changes the subspace on which each frame acts rather than fine-tuning a random frame. Second, frames barely change across patch positions, consistent with the static and modulated variants performing alike (Table 14). Third, training creates pairwise structure: the spread of a_{ij} grows about fourfold, and a_{ij} becomes positively rank-correlated with the correlation of the training series, while road-adjacent sensors become slightly more aligned than distant ones. These associations are modest. Together with the per-pair audit in Appendix E and the transplant results in Appendix D, they indicate that frames encode a data-dependent, partly correlation-aligned convention per variable that the surrounding modules co-adapt to, rather than a readily interpretable map of known relationships such as spatial proximity.

H.7 Scope of the Evaluation

Routing rules in Figure 1 range from uninformative (a fixed random stochastic matrix) to static and data-driven (row-normalized absolute Pearson correlation of the training series) to instance-adaptive (a cosine-similarity softmax and dot-product attention). The interface also transfers to a published cross-variable model: inserted into iTransformer's variate attention, it reduces PEMSD4 error by 47.1% (Appendix G.2). We did not test graph-learning routers or insertion into CARD, whose channel-aligned attention already modifies value processing, and we did not try Cayley or matrix-exponential orthogonal parameterizations; the fixed Hadamard, affine, and relation-operator controls cover other construction choices. Long-horizon baselines were run on six datasets (Table 12); on PEMSD8 and PEMSD4 we report only Frameformer's horizon curve (Appendix F).

H.8 Theory-Matched Synthetic Test

We generate $N = 12$ variables of length 12,000 as lagged noisy copies of one driver, $x_j(t) = s(t - \tau_j) + 0.3\,\epsilon_j(t)$, where s is Gaussian-smoothed white noise plus a period-48 cycle. In the *synchronous* regime all $\tau_j = 0$, so the ideal transfer is common to all pairs; in the *lagged* regime the τ_j are spread evenly over 0 to 60 steps, so the transfer depends on the pair. We train Frameformer with the benchmark configuration ($h = 96$, 7/1/2 split) with identity, fixed random, or learned frames, over 3 seeds.

The results in Table 4 match the prediction of Appendix H.1. When the ideal transfer is common, frames give no gain. When it is lag-dependent, shared-space mixing is worse than in the synchronous case, learned frames reduce error by 18%, and fixed random frames help only slightly (3%). This separates the two roles seen on real data: a persistent convention alone helps little when the needed transfer is structured, and learning the frames is what recovers it. We do not claim that the learned $\mathbf{R}_{ij}$ equal the ground-truth rotations, since the latent coordinates are learned and identified only up to a common gauge.